**ESERCIZIO Scheduling 1**

In un sistema vengono generati 5 processi (A,B,C,D,E), con durate (in millisecondi) sotto specificate:

|  |  |
| --- | --- |
| Processo | Durata |
| A | 25 |
| B | 60 |
| C | 5 |
| D | 15 |
| E | 10 |

Tutti i processi avanzano senza mai sospendersi.

Calcolare il tempo di permanenza nel sistema di ogni processo (definito come differenza tra il tempo di completamento dell’esecuzione e il tempo di arrivo nel sistema,) e il tempo medio di permanenza dei processi, con le seguenti politiche di scheduling

1. Politica FIFO;

2. Politica SJF (Shortest Job First).

Nel calcolo dei tempi di permanenza, si ignori il tempo di commutazione di contesto.

**Soluzione**

1. Politica FIFO:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROCESSO | TEMPO DI ARRIVO | INIZIA  ESECUZIONE | TERMINA ESECUZIONE | TEMPO DI PERMANENZA NEL SISTEMA |
| A | 0 | 0 | 25 | 25 |
| B | 0 | 25 | 25+60=85 | 85 |
| C | 0 | 85 | 85+5=90 | 90 |
| D | 0 | 90 | 90+15=105 | 105 |
| E | 0 | 105 | 105+10=115 | 115 |

Tempo medio di permanenza: (25+85+90+105+115)/5=84 msec

2. Politica SJF:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROCESSO | TEMPO DI ARRIVO | INIZIA  ESECUZIONE | TERMINA ESECUZIONE | TEMPO DI PERMANENZA NEL SISTEMA |
| C | 0 | 0 | 5 | 5 |
| E | 0 | 5 | 5+10=15 | 15 |
| D | 0 | 15 | 15+15=30 | 30 |
| A | 0 | 30 | 30+25=55 | 55 |
| B | 0 | 55 | 55+60=115 | 115 |

Tempo medio di permanenza: (5+15+30+55+115)/5=44 msec

**ESERCIZIO Scheduling 2**

In un sistema vengono generati 5 processi (A,B,C,D,E), con i tempi di arrivo e le durate (in millisecondi) sotto specificate:

|  |  |  |
| --- | --- | --- |
| Processo | Durata | Tempo di arrivo |
| A | 25 | 0 |
| B | 60 | 1 |
| C | 5 | 2 |
| D | 15 | 3 |
| E | 10 | 4 |

Tutti i processi avanzano senza mai sospendersi.

Calcolare il tempo di *permanenza nel sistema* di ogni processo (definito come differenza tra il tempo di completamento dell’esecuzione e il tempo di arrivo nel sistema) con le seguenti politiche di scheduling

1. Politica FIFO
2. Politica Round-Robin con quanto di tempo pari a 5 msec.

In entrambi i casi si ignori il tempo di commutazione di contesto.

**Soluzione**

1. Politica FIFO:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROCESSO | TEMPO DI ARRIVO | INIZIA  ESECUZIONE | TERMINA ESECUZIONE | TEMPO DI PERMANENZA NEL SISTEMA |
| A | 0 | 0 | 25 | 25 |
| B | 1 | 25 | 25+60=85 | 85-1=84 |
| C | 2 | 85 | 85+5=90 | 90-2=88 |
| D | 3 | 90 | 90+15=105 | 105-3=102 |
| E | 4 | 105 | 105+10=115 | 115-4=111 |

1. Round Robin:

t= 0 25 50 75 100 125 150 175

Terminato

D

E

A

B

C

Terminato

Terminato

Terminato

Terminato

# Tempo di *permanenza nel sistema*:

A =75

B =115-1=114

C =15-2=13

D = 60-3=57

E = 45-4=41

**ESERCIZIO Scheduling 3**

In un sistema vengono generati 6 processi (A,B,C,D,E,F), con i tempi di arrivo, le priorità e le durate (in millisecondi) sotto specificate:

|  |  |  |  |
| --- | --- | --- | --- |
| PROCESSO | TEMPO DI ARRIVO | PRIORITÀ | DURATA |
| A | 0 | 2 | 10 |
| B | 8 | 1 | 24 |
| C | 18 | 3 | 6 |
| D | 28 | 4 | 10 |
| E | 32 | 2 | 11 |
| F | 36 | 3 | 7 |

Lo scheduling del processore avviena con una politica a prorità (che assegna il processore al processo che ha il valore più elevato di priorità e a pari priorità, al processo arrivato per primo) e con prerilascio. Si suppone che, una volta in esecuzione, ogni processo avanzi senza mai sospendersi.

Riempire la seguente tabella, utilizzando una riga per ogni evento che provoca la riassegnazione del processore e specificando, in ogni riga, il processo in esecuzione e la composizione della coda pronti subito dopo il verificarsi dell’evento.

**SOLUZIONE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TEMPO | EVENTO | PROC. IN ESEC. | CODA PRONTI | NOTE |
| 0 | Arriva A | A | Ø |  |
| 8 | Arriva B | A | B |  |
| 10 | Terrmina A | B | Ø |  |
| 18 | Arriva C | C | B | B: tempo residuo 16 |
| 24 | Terrmina C | B | Ø |  |
| 28 | Arriva D | D | B | B: tempo residuo 12 |
| 32 | Arriva E | D | E 🡪 B |  |
| 36 | Arriva F | D | F 🡪 E 🡪 B |  |
| 38 | Terrmina D | F | E 🡪 B |  |
| 45 | Terrmina F | E | B |  |
| 56 | Terrmina E | B | Ø |  |
| 68 | Terrmina B | Ø | Ø |  |

**ESERCIZIO Scheduling 4**

In un sistema vengono generati 5 processi (A,B,C,D,E), con i tempi di arrivo e le durate (in millisecondi) sotto specificate:

|  |  |  |
| --- | --- | --- |
| Processo | Durata | Tempo di arrivo |
| A | 45 | 0 |
| B | 55 | 13 |
| C | 15 | 21 |
| D | 5 | 25 |
| E | 15 | 29 |

Si suppone che tutti i processi avanzino senza mai sospendersi.

Lo scheduler adotta la politica *Shortest Remaining Time First (SRTT),* la qualeseleziona per l’esecuzione il processo con minor tempo residuo in esecuzione e prevede il prerilascio.

Calcolare il tempo di *permanenza nel sistema* di ogni processo (definito come differenza tra il tempo di completamento dell’esecuzione e il tempo di arrivo nel sistema)

**SOLUZIONE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tempo t= | Evento | Processo in esecuzione | Tempo residuo processo in esecuzione | Coda pronti (con tempo residuo di esecuzione) |
| 0 | Generato processo A | A | 45 | - |
| 13 | Generato processo B | A | 32 | B(55) |
| 21 | Generato processo C | C | 15 | A(24), B(55) |
| 25 | Generato processo D | D | 5 | C(11), A(24), B(55) |
| 29 | Generato processo E | D | 1 | C(11), E(15),A(24), B(55) |
| 30 | Termina D | C | 11 | E(15),A(24), B(55) |
| 41 | Termina C | E | 15 | A(24), B(55) |
| 56 | Termina E | A | 24 | B(55) |
| 80 | Termina A | B | 55 | - |
| 135 | Termina B | - | - | - |

Di conseguenza:

|  |  |  |  |
| --- | --- | --- | --- |
| Processo | Tempo di arrivo nel sistema | TERMINA ESECUZIONE | TEMPO DI PERMANENZA NEL SISTEMA |
| A | 0 | 80 | 80 |
| B | 13 | 135 | 122 |
| C | 21 | 41 | 20 |
| D | 25 | 30 | 5 |
| E | 29 | 56 | 27 |

# ESERCIZIO Scheduling 5

Un sistema gestisce il processore con politica RoundRobin con quanto di tempo di *5 msec*. Quando un processo va in esecuzione gli viene assegnato un intero quanto di tempo di *5 ms*, indipendentemente dal tempo consumato nel precedente turno di esecuzione.

Nel sistema sono presenti 4 processi (A,B,C,D) e un semaforo di mutua esclusione *mux*. Al tempo t passa in esecuzione il processo A, la coda pronti contiene i processi B->C->D e il semaforo *mux* ha valore 0 e coda vuota.

Si chiede quale è il processo in esecuzione e la composizione della coda pronti *nell’intervallo di tempo da t a t+20,* nel corso del quale si verificano i seguenti eventi:

1. il processo in esecuzione si sospende al tempo t+8 sul semaforo *mux*;
2. il processo in esecuzione si sospende al tempo t+14 sul semaforo *mux*;
3. il processo in esecuzione esegue una *signal*(*mux*) al tempo t+16

# SOLUZIONE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tempot+ ….. | Evento | In Esecuzione | Coda Pronti | Sospesi su *mux* |
| 0 | A in esecuzione | A | B, C, D | - |
| 5 | Scad. quanto di tempo | B | C,D,A | - |
| 8 | B sospeso su *mux* | C | D,A | B |
| 13 | Scad. quanto di tempo | D | A,C | B |
| 14 | D sospeso su *mux* | A | C | B,D |
| 16 | Riattivato B | A | C,B | D |
| 19 | Scad. quanto di tempo | C | B,A | D |
| 20 | Situazione a t+20 | C | B,A | D |

**ESERCIZIO Scheduling 6**

Un sistema gestisce il processore combinando le politiche a priorità e RoundRobin con la tecnica delle code multiple (una coda FIFO per ogni valore di priorità; i processi pronti di uguale priorità sono inseriti in una stessa coda; il processore viene assegnato al processo che occupa la prima posizione nella coda non vuota di massima priorità; ai processi pronti di uguale priorità si applica la politica Round Robin).

Il quanto di tempo è di *10 msec*.

La politica prevede il prerilascio, che avviene immediatamente dopo l’evento che lo provoca, senza attendere l’esaurimento del quanto di tempo corrente. Quando un processo va in esecuzione gli viene assegnato un intero quanto di tempo, indipendentemente dal tempo consumato nel precedente turno di esecuzione.

Al tempo T, nel sistema sono presenti i seguenti processi:

* Processo A, con priorità 2, che al tempo T è in stato di attesa sul semaforo Sem1;
* Processo B, con priorità 3, che al tempo T è in stato di attesa sul semaforo Sem1;
* Processo C, con priorità 1, che al tempo T è in stato di pronto;
* Processo D, con priorità 2, che al tempo T passa in stato di esecuzione.
* Processo E, con priorità 1, che al tempo T è in stato di pronto;

Al tempo T la coda corrispondente alla priorità 1 contiene: C->E (C è in testa), e tutte le altre code sono vuote. La coda del semaforo Sem1 contiene invece A->B (A è in testa).

Si chiede quale è il processo in esecuzione e la composizione delle 3 code al tempo T+30 se si verifica la seguente sequenza di eventi:

1. al tempo T+8 il processo in esecuzione esegue una *signal* sul semaforo Sem1;
2. al tempo T+12 il processo in esecuzione esegue una *wait* sul semaforo Sem1;
3. al tempo T+14 termina il processo in esecuzione;
4. al tempo T+26 il processo in esecuzione esegue una *wait*  sul semaforo Sem1.

**SOLUZIONE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tempo | Evento | In esecuzione | Coda  priorità 1 | Coda  priorità 2 | Coda  priorità 3 | Sem1:  <valore,coda> |
| T | D D in esecuzione | D | C->E |  |  | <0,A->B> |
| T+8 | Riattivato A, pr. 2 | D | C->E | A |  | <0,B> |
| T+10 | Quanto di tempo | A | C->E | D |  | <0,B> |
| T+12 | A sospeso su Sem1 | D | C->E |  |  | <0,B->A> |
| T+14 | D termina | C | E |  |  | <0,B->A> |
| T+24 | Quanto di tempo | E | C |  |  | <0,B->A> |
| T+26 | E sospeso su Sem1 | C |  |  |  | <0,B->A->E> |
| T+30 | Situazione a T+30 | C |  |  |  | <0,B->A->E> |

**ESERCIZIO Scheduling-7**

Un sistema gestisce il processore con una politica a code multiple, con 5 classi di priorità di valori 0, 1, 2, 3 e 4. I processi pronti sono inseriti nella coda della classe corrispondente alla propria priorità; il processore viene assegnato al processo che occupa la prima posizione nella coda non vuota di massima priorità; ai processi pronti della stessa classe di priorità si applica la politica Round Robin con quanto di tempo (QdT) di *10 msec*. Quando un processo passa in esecuzione, gli viene assegnato un intero quanto di tempo, indipendentemente dal tempo consumato nel precedente turno di esecuzione.

La politica prevede la revoca del processore, che avviene *immediatamente* al verificarsi di uno dei seguenti eventi:

* il processo in esecuzione esaurisce il quanto di tempo;
* viene riattivato un processo con priorità maggiore di quello in esecuzione.

La priorità di ogni processo Pi  è definita dinamicamente nel modo seguente:

* inizialmente si assegna un valore *normale*, nell’intervallo [0, 4];
* se Pi è in stato di pronto ed è rimasto ininterrottamente in questo stato per almeno 3 turni di esecuzione da parte di altri processi, ciascuno della durata di un intero quanto di tempo, la priorità di Pi viene transitoriamente elevata al valore 4. Questo valore permane fino a quando il processo Pi ha ottenuto due turni di esecuzione, utilizzandoli per l’intero quanto di tempo o anche parzialmente, dopo di che la priorità torna al valore normale. Si noti che, ai fini dell’innalzamento della priorità, non si considerano i turni di esecuzione di altri processi nei quali il processo in esecuzione si sospende prima dell’esaurimento del quanto di tempo.

Al tempo *t* sono presenti i seguenti processi:

* Processo A, con priorità 3, in stato di esecuzione;
* Processo B, con priorità 1, in stato di pronto;
* Processo C, con priorità 2, in stato di attesa sul semaforo Sem1, la cui coda non contiene altri processi;
* Processo D, con priorità 2, in stato di attesa sul semaforo Sem2, la cui coda non contiene altri processi;

Sono definiti i semafori Sem1, Sem2 e Sem3, con i seguenti valori e composizioni delle code:

* Sem1: Valore 0, CodaSem1 🡪 C
* Sem2: Valore 0, CodaSem2 🡪 D
* Sem3: Valore 0, CodaSem3 🡪 ∅

Al tempo *t* si ha la seguente situazione:

* Processo A: il contatore di programma punta alla chiamata della primitiva *wait(Sem3)* che ha tempo di esecuzione 0; successivamente il programma invoca la funzione *FunzioneA;*
* Processo B: il contatore di programma punta alla chiamata della primitiva *signal(Sem1)* che ha tempo di esecuzione 0; successivamente il programma invoca la funzione *FunzioneB,* la cui esecuzione richiede *15 msec;* successivamante invoca la primitiva *signal(Sem3)* che ha tempo di esecuzione 0 , e quindi invoca nuovamente la funzione *FunzioneB.*
* Processo C: il contatore di programma punta all’invocazione della funzione *FunzioneC1* la cui esecuzione richiede *5 msec,*  successivamente invoca la primitiva *signal(Sem2)* che ha tempo di esecuzione 0; quindi invoca la funzione *FunzioneC2,* la cui esecuzione richiede *50 msec;*
* Processo D: il contatore di programma punta all’invocazione della funzione *FunzioneD* la cui esecuzione richiede *60 msec.*

Utilizzando la tabella seguente, si chiede di analizzare l’evoluzione dello stato dei processi A, B, C, D in base alla politica di scheduling, fino al tempo *t+50.*

**SOLUZIONE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tempo | Evento | Processo A | Processo B | Processo C | Processo D |
|  |  |  |  |  |  |
| t+0 | A esegue *wait(Sem3)* | Stato: attesa su Sem3 | Stato: esecuzione | Stato: attesa su Sem1 | Stato: attesa su Sem2 |
| Priorità: 3 | Priorità: 1 | Priorità: 2 | Priorità: 2 |
| QdT in stato di pronto: 0 | QdT in stato di pronto:0 | QdT in stato di pronto:0 | QdT in stato di pronto:0 |
|  |  |  |  |  |  |
| t+0 | B esegue *signal(Sem1)* | Stato: attesa su Sem3 | Stato: pronto | Stato: esecuzione | Stato: attesa su Sem2 |
| Priorità: 3 | Priorità: 1 | Priorità: 2 | Priorità: 2 |
| QdT in stato di pronto: 0 | QdT in stato di pronto:0 | QdT in stato di pronto:0 | QdT in stato di pronto:0 |
|  |  |  |  |  |  |
| t+5 | C esegue *signal(Sem2)* | Stato: attesa su Sem3 | Stato: pronto | Stato: esecuzione | Stato: pronto |
| Priorità: 3 | Priorità: 1 | Priorità: 2 | Priorità: 2 |
| QdT in stato di pronto: 0 | QdT in stato di pronto:0 | QdT in stato di pronto:0 | QdT in stato di pronto:0 |
|  |  |  |  |  |  |
| t+10 | Esaurito QdT | Stato: attesa su Sem3 | Stato: pronto | Stato: pronto | Stato: esecuzione |
| Priorità: 3 | Priorità: 1 | Priorità: 2 | Priorità: 2 |
| QdT in stato di pronto: 0 | QdT in stato di pronto:1 | QdT in stato di pronto:0 | QdT in stato di pronto:0 |
|  |  |  |  |  |  |
| t+20 | Esaurito QdT | Stato: attesa su Sem3 | Stato: pronto | Stato: esecuzione | Stato: pronto |
| Priorità: 3 | Priorità: 1 | Priorità: 2 | Priorità: 2 |
| QdT in stato di pronto: 0 | QdT in stato di pronto:2 | QdT in stato di pronto:0 | QdT in stato di pronto:0 |
|  |  |  |  |  |  |
| t+30 | Esaurito QdT | Stato: attesa su Sem3 | Stato: esecuzione | Stato: pronto | Stato: pronto |
| Priorità: 3 | Priorità: 4 | Priorità: 2 | Priorità: 2 |
| QdT in stato di pronto: 0 | QdT in stato di pronto:3 | QdT in stato di pronto:0 | QdT in stato di pronto:1 |
|  |  |  |  |  |  |
| t+40 | Esaurito QdT | Stato: attesa su Sem3 | Stato: esecuzione | Stato: pronto | Stato: pronto |
| Priorità: 3 | Priorità: 4 | Priorità: 2 | Priorità: 2 |
| QdT in stato di pronto: 0 | Turni di esec:1 | QdT in stato di pronto:0 | QdT in stato di pronto:2 |
|  |  |  |  |  |  |
| t+45 | B esegue *signal(Sem3)* | Stato: pronto | Stato: esecuzione | Stato: pronto | Stato: pronto |
| Priorità: 3 | Priorità: 4 | Priorità: 2 | Priorità: 2 |
| QdT in stato di pronto: 0 | Turni di esec:1 | QdT in stato di pronto:1 | QdT in stato di pronto:2 |
|  |  |  |  |  |  |
| T+50 | Esaurito QdT | Stato: esecuzione | Stato: pronto | Stato: pronto | Stato: esecuzione |
|  |  | Priorità: 3 | Priorità: 1 | Priorità: 2 | Priorità: 4 |
|  |  | QdT in stato di pronto: 0 | Turni di esec:2 | QdT in stato di pronto:2 | QdT in stato di pronto:3 |